Optimization-Based Inverse Model of Soft Robots With Contact Handling

Eulalie Coevoet, Adrien Escande, Christian Duriez

Optimization-based inverse model of Soft Robots with Contact Handling

Eulalie Coevoet[1], Adrien Escande[2] and Christian Duriez[1]

***Abstract*— This paper presents a physically-based algorithm to interactively simulate and control the motion of soft robots interacting with their environment. We use the Finite Element Method (FEM) to simulate the non-linear deformation of the soft structure, its actuators, and surroundings, and propose a control method relying on a quadratic optimization to find the inverse of the model. The novelty of this work is that the deformations due to contacts, including self-collisions, are taken into account in the optimization process. We propose a dedicated and efficient solver to handle the linear complementarity constraints introduced by the contacts. Thus, the method allows interactive transfer of the motion of soft robots from their task space to their actuator space while interacting with their surrounding. The method is generic and tested on several numerical examples and on a real cable-driven soft robot.**

I. INTRODUCTION

Soft robotics is an emerging opportunity to re-think the way robots are designed, used, and controlled, and to provide new capabilities. Using soft materials or flexible structures, these robots take motion by deformation. In opposition to articulated rigid structures, the kinematics do not only depend on the geometry, but also on the material properties. Moreover, they have a theoretical infinite number of degrees of freedom and it is hard to get an analytic model that describes the mechanical behavior with accuracy. These characteristics make their modeling and control more complex, particularly when the robot is being deformed by contact with its environment. This issue has been identified as a challenging open problem by several reviews on soft robotics [1] [2] [3]. In this work, we propose the first generic method for obtaining the inverse model of a soft robot in contact with a known environment. We use the Finite Element Method (FEM) for modeling the deformations and Lagrangian constraints for actuators. The inverse model is obtained by optimization and allows interactive control of the motion.

Soft robots are usually inspired by living organisms, such as snakes, octopi, or caterpillars that have a continuously highly deformable structure. The compliance of these bodies is particularly suitable for exploration and manipulation in cluttered and sensitive environments. Using large strain deformation, they can reduce or extend their nominal dimensions, bend, and adapt their shape to the environment. This compliance in the interaction reduces risks of damage for both the robot and its surrounding. This makes them ideal for safe interaction with human beings and especially for use in medical applications such as surgery. Therefore, it is essential to propose a control approach that takes into account possible collisions and adjusts the deformations needed to reach a desired position. Collisions and contacts handling are very challenging because they significantly increase the complexity of the control. In this paper, we propose an algorithm that includes contacts in the optimization process and a specific solver to compute the resolution at interactive rates (between 30Hz and 100Hz).

[1]INRIA and University of Lille, France

[2]CNRS-AIST Joint Robotic Laboratory (JRL) UMI3218/RL, Tsukuba, Japan

*Adrien Escande is supported by the Airbus Group-AIST-CNRS Joint Research Program.

II. RELATED WORKS

A. Soft robot motion control

Several difficulties are experienced in the modeling and control of soft robots: First, the number of actuators is finite while the number of degrees of freedom is infinite. Therefore, many DoFs of the robot are not directly controllable. Second, the actuators can also be redundant and several possible deformable postures can lead to the same position of the end effector of the robot. Third, many *new* parameters (material properties, external loads, weight...) have an influence on the posture of the robot.

The kinematics of soft robots has been studied in the particular case of continuum robots [4] [5]. The authors use the hypothesis of a piece-wise constant curvature to keep a geometrical approach to the kinematics and simplify the problem. The approach is then based on defining the bending deformation of soft segments as curvatures. But this hypothesis requires a specific design of the robot and no external and local load. For soft robots with non-continuous curvature, a first approach based on the mechanical modeling and a real-time FEM simulation of the structure and its actuators was proposed in [6] [7]. Our paper extends these works by using a specific solver to handle interactions and self-collisions in the simulation. In this paper, the inverse problem is formulated as a Quadratic Program (QP) with linear complementarity constraints.

B. Inverse model of deformation with contact handling

Inverse FEM simulation with contact handling has been used in a few works in computer graphics, to control motion of actuated virtual soft bodies [8] [9] [10] [11]. In [8] and [10] the authors propose to control simulated skeleton-driven deformable characters in near real-time. They use a penalty-based contact model to simulate collision. However, in some cases, in order to make the character use the ground for push off, they temporarily create a hard constraint between a bone and the ground. In our work, we made no assumption on the characteristics of a contact, allowing each contact to be used for the actuation. We use Signorini's law

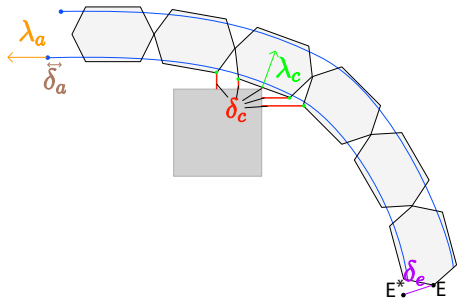

Fig. 1: Illustration of the quantities of eq. 3 for a deformable robot with cable actuation (in blue), in the presence of an obstacle (grey square). E^* is the desired position for the controlled point E.

to model the collision and guarantee a physical response. While more general, this makes the problem harder to solve in real-time. In [11], the authors use FEM and muscle fibers actuation to simulate a soft body character's locomotion. The solver we propose in this paper to handle the complementarity constraints introduced by Signorini's law, is similar to theirs. However, in this study, we propose to find the inverse of the model of real robots. The method has real-time performance thanks to, among others, a projection of the model in the space of the optimization variables. To the best of our knowledge, this paper is the first work in soft robotic to propose an interactive motion control in the task space which handles interactions with the environment.

III. METHOD

A. Inverse problem formulation

The goal is to find how to actuate the robot so that selected points reach desired positions. We use the FEM implementation provided by the framework Sofa [12] and a tetrahedral or hexahedral mesh representation of the robot body to simulate the deformation of the volume structure. The tetrahedral volumetric meshes are generated with the software Gmsh or the CGALPlugin of Sofa.

The configuration of the robot at a given time is obtained by solving the static equilibrium between the internal forces of the deformable structure $f(x)$, the external loads f_{ext} (such as gravity) and the contributions of actuators $H_a^T\lambda_a$ and contacts $H_c^T\lambda_c$, yielding:

$$f(x) = -f_{ext} - H_a^T\lambda_a - H_c^T\lambda_c, \tag{1}$$

with H^T the direction of the effort applied by the constraints (actuators and contacts) on the FEM nodes and λ the intensity of this effort. At each time step i of the simulation, we compute a linearization of the internal forces:

$$f(x_i) \approx f(x_{i-1}) + K(x_{i-1})dx \tag{2}$$

where $K(x)$ is the tangent stiffness matrix that depends on the current position of the FEM nodes, and dx is the difference between consecutive positions in time $dx = x_i - x_{i-1}$. The size of the matrix K is related to the number of FEM nodes, which is often very large. An optimization in the motion space would then be computationally expensive. Instead, we project the model in the space of the actuation and contact variables, leading to (see [6] for more details):

$$\begin{bmatrix}\delta_e\\ \delta_a\\ \delta_c\end{bmatrix} = \begin{bmatrix}W_{ea} & W_{ec}\\ W_{aa} & W_{ac}\\ W_{ca} & W_{cc}\end{bmatrix}\begin{bmatrix}\lambda_a\\ \lambda_c\end{bmatrix} + \begin{bmatrix}\delta_e^{\text{free}}\\ \delta_a^{\text{free}}\\ \delta_c^{\text{free}}\end{bmatrix} \tag{3}$$

$$\delta_{max} \geq \delta_a \geq \delta_{min} \tag{4}$$

$$\lambda_{max} \geq \lambda_a \geq \lambda_{min} \tag{5}$$

$$0 \leq \lambda_c \perp \delta_c \geq 0 \tag{6}$$

where matrices $W_{ij} = H_i K^{-1} H_j^T$ $(i,j = e,a,c)$ are homogeneous to a compliance, and gather the mechanical coupling between effector points e, actuators a and contacts c. We thus have a relation between the actuator and contact forces, λ_a and λ_c, the shift between controlled points and their desired positions δ_e, the displacement (or volume growth for pneumatic actuation) of actuator δ_a and the gap between two colliding points δ_c (see Fig. 1). Eq. (4) and (5) respectively are, constraints on actuators such as limits on the cables displacements, and limits on actuation forces. In this work, we follow Signorini's law for contact (6). We do not consider friction. This law guaranties that there is no interpenetration and that the contact forces are well oriented. The values δ_e^{free}, δ_a^{free} and δ_c^{free} respectively corresponds to the shift δ_e, the displacement (or volume growth) δ_a, and the interpenetration δ_c computed during a free motion, that is when solving the structure with no effort on actuators and contacts, i.e $\lambda_a = 0$ and $\lambda_c = 0$ (see [6]). We note n_a and n_c the number of actuator and contact forces.

To control the robot, we want to find the actuation λ_a so that effectors reach their desired positions. This corresponds to minimizing the norm $\|\delta_e\|$, while respecting the constraints (4), (5), and (6). Using eq. (3) to get the expressions of δ_e and δ_c, we formulate the following Quadratic Program with (linear) Complementarity Constraints (QPCC):

$$\begin{aligned}\min_{\lambda_c,\lambda_a}\ & \left\|W_{ec}\lambda_c + W_{ea}\lambda_a + \delta_e^{\text{free}}\right\|^2 \\ \text{s.t. } & A\begin{bmatrix}\lambda_a\\ \lambda_c\end{bmatrix} \geq b \\ & 0 \leq \lambda_c \perp W_{cc}\lambda_c + W_{ca}\lambda_a + \delta_c^{\text{free}} \geq 0\end{aligned} \tag{7}$$

When no potential collision has been detected ($n_c = 0$), the problem is a simple QP. Note that λ_c is part of the optimization variables, which allows the controller to make use of contact forces to achieve the desired motion.

If the robot has a number of actuators greater than the DoFs of the controlled points, the QPCC may have an infinite number of solutions. In such case, we introduce in the minimization expression, the mechanical work of the actuator forces $E = \Delta\delta_a\lambda_a$, with $\Delta\delta_a = \delta_a - \delta_a^{free}$ the displacements of the actuators during a time step. E is linked to the mechanical energy of the robot deformation. We then minimize the sum $(||\delta_e||^2 + \epsilon E)$, with ϵ chosen sufficiently small so that the deformation energy does not

disrupt the quality of the controlled points positioning. In the implementation, we use $\epsilon = 1e^{-3}||W_{ea}^T W_{ea}||_\infty / ||W_{aa}||_\infty$ (with the norm $||.||_\infty$ being the maximum absolute row sum of the matrix). A unique solution of the problem can then be found, without a significant impact on the quality of the solution. In the rest of the paper, we omit this damping term ϵE, for the sake of clarity. It is straightforward to extend the developments below to incorporate it.

B. Quadratic program with linear complementarity constraint solver

Recent works in optimization have addressed the problem of linear and quadratic programs with linear complementarity constraints [13] [14]. They seek to find the global optimum of the problem and demonstrate that it can be accomplished in finite time. However, as finding the global minimum is difficult to achieve in real-time, we developed our own specific solver based on the decomposition method as mentioned in [15].

The complementarity constraints (6) defines 2^{n_c} choices. Each of them can be characterized by a subset I of $\{1, \ldots, n_c\}$ giving the elements of λ_c that are forced to be zero. Let e_i be the i-th column of the n_c-by-n_c identity matrix, and define S_I as the matrix whose columns are the e_i for i in I. Given a matrix M, MS_I (respectively $S_\mathrm{I}^T M$) selects the columns (respectively rows) of M indexed by I. Likewise, we define $\bar{S}_\mathrm{I}$ for i not in I. $\begin{bmatrix} S_\mathrm{I} & \bar{S}_\mathrm{I} \end{bmatrix}$ is a permutation (and thus orthonormal) matrix. Given a complementarity choice I, QPCC (7) rewrites

$$\min_{\lambda_a, \lambda_c} \ \left\| W_{ea}\lambda_a + W_{ec}\lambda_c + \delta_e^{free} \right\|^2 \tag{8}$$

$$\text{s.t. } A\lambda_a \geq b \tag{9}$$

$$S_\mathrm{I}^T \lambda_c = 0 \tag{10}$$

$$S_\mathrm{I}^T (W_{cc}\lambda_c + W_{ca}\lambda_a + \delta_c^{\mathrm{free}}) \geq 0 \tag{11}$$

$$\bar{S_\mathrm{I}}^T \lambda_c \geq 0 \tag{12}$$

$$\bar{S_\mathrm{I}}^T (W_{cc}\lambda_c + W_{ca}\lambda_a + \delta_c^{\mathrm{free}}) = 0 \tag{13}$$

This is a QP piece of (7) we refer to as QP_I.

We propose an iterative method that starts from an initial feasible set I. After solving QP_I, we inspect the state of the inequality constraints. If an inequality constraint has reached its boundary at the end of the optimization, it means that the solution may potentially be further optimized by pivoting the corresponding constraint. We set each inequality constraints that reached their boundary at the end of the optimization as candidate for pivot. To guarantee the convergence of the algorithm only one constraint can be pivoted at a time. As mentioned in [15], one way to determine which constraint should be the best candidate for pivot is to examine the values of the dual variables. In our implementation, we select the candidate with the greater dual variable. By pivoting the corresponding complementarity constraint we get a new QP with a different set of linear constraints. We solve this new problem and repeat the process until there is no more candidate for pivot, effectively solving a sequence of QP_I. A proof of the convergence of this kind of algorithm to a stationary point is given in [16].

C. Reduced formulation

The above scheme can be improved by taking into account the specificity of our problem. Indeed, W_{cc} is positive definite (because K is positive definite). As a result (see below equations), λ_c is an affine function of λ_a for a given I. This allows the removal of λ_c from QP_I to solve a smaller problem. We name this the *reduced formulation*. In the remainder of this section, we drop the index I for matrices S_I and $\bar{S}_\mathrm{I}$.

Using that $\lambda_c = \begin{bmatrix} S & \bar{S} \end{bmatrix} \begin{bmatrix} S & \bar{S} \end{bmatrix}^T \lambda_c = SS^T\lambda_c + \bar{S}\bar{S}^T\lambda_c$, we get from eq. (10) that $\lambda_c = \bar{S}\bar{S}^T\lambda_c$. Reintroducing this result in eq. (13), and solving for $\bar{S}^T\lambda_c$ yields:

$$\bar{S}^T\lambda_c = -\left(\bar{S}^T W_{cc}\bar{S}\right)^{-1} \bar{S}^T \left(W_{ca}\lambda_a + \delta_a^{\mathrm{free}}\right)$$

We define A_I and b_I such that the above relation rewrites $\bar{S}^T\lambda_c = A_\mathrm{I}\lambda_a + b_\mathrm{I}$. Then we have the affine relation

$$\lambda_c = \bar{S}\left(A_\mathrm{I}\lambda_a + b_\mathrm{I}\right) \tag{14}$$

and QP_I is equivalent to the following QP we name QP_I^r:

$$\begin{aligned} \min_{\lambda_a} \ & \left\| \left(W_{ec}\bar{S}A_\mathrm{I} + W_{ea}\right)\lambda_a + W_{ec}\bar{S}b_\mathrm{I} + \delta_e^{free} \right\|^2 \\ \text{s.t. } & A\lambda_a \geq b \\ & A_\mathrm{I}\lambda_a \geq b_\mathrm{I} \\ & S^T\left(W_{cc}\bar{S}A_\mathrm{I} + W_{ca}\right)\lambda_a + S^T\left(W_{cc}\bar{S}b_\mathrm{I} + \delta_e^{\mathrm{free}}\right) \geq 0 \end{aligned} \tag{15}$$

We obtain QP_I^r by solving eqs. (10) and (13), what would have been done anyway by the QP solver. But we did so by leveraging the structure of the problem (with respect to λ_a and λ_c), and we can take into account the fact that $\bar{S}^T W_{cc}\bar{S}$ is positive definite[1]. In particular, we can use the Cholesky decomposition of $\bar{S}^T W_{cc}\bar{S}$ to compute its inverse[2], which is much cheaper computationally than the more general decomposition the QP solver would need to use. As a result, given the matrices W_{ij} and A, and the vectors δ_i^{free} and b, solving QP_I is slower than computing the matrices and vectors in QP_I^r and solving QP_I^r. The ratio between the two computation times depends on n_a and n_c. When $n_a >> n_c$, the timings are the similar (in particular, if $n_c = 0$, both problems are the same). However, for a fixed n_a, the reduced formulation becomes better and better as n_c increases. For example, for $(n_a, n_c) = (3, 3)$, the average ratio is 1.25. It is 1.88 for $(3, 10)$, and 7.82 for $(3, 50)$. Usually, n_c is much larger than n_a so that the reduced formulation has a real computational advantage.

The computation time could be further reduced when we consider the sequence of resolutions performed in the iterative method described above. Indeed, in this case, two successive set I differ by only one element, so that we pass

[1]It is a principal minor of the positive definite matrix W_{cc}.

[2]Note that, following the recommended practice, we do not compute explicitly the inverse. Rather (see [17]), given the Cholesky decomposition $\bar{S}^T W_{cc}\bar{S} = LL^T$, with L a lower triangular matrix, we compute $\begin{bmatrix} A_\mathrm{I} & b_\mathrm{I} \end{bmatrix} = -L^{-T}L^{-1}\bar{S}^T \begin{bmatrix} W_{ca} & \delta_a^{\mathrm{free}} \end{bmatrix}$, where the left multiplication by L^{-1} and L^{-T} are obtained by forward and backward substitution.

from one matrix $\bar{S}^T W_{cc} \bar{S}$ to an other by adding or removing one row and one column. Therefore, there is no need to compute the Cholesky decomposition from scratch at each iteration, but it can rather be updated. The full decomposition is in $O(k^3)$ where k is the size of the matrix, while the update is in $O(k^2)$[17]. Preliminary tests show that this reduces the computation time by $20-25\%$ for all but the first iteration. We do not use this improvement yet in the controller.

D. Initialization

As mentioned above, our solver requires an initial feasible set I. In the implementation, this initial set is found by solving the contacts as a Linear Complementarity Constraint (LCP) while considering the actuator force λ_a constant:

$$\begin{aligned} \delta_c &= W_{cc}\lambda_c + W_{ca}\lambda_a + \delta_c^{free} \\ 0 \leq \lambda_c &\perp \delta_c \geq 0 \end{aligned} \tag{16}$$

This system has a solution for any λ_a because W_{cc} is positive definite[18]. Thus there is always at least one feasible set I. Furthermore, the solution is unique, and is given explicitly by eq. (14). The initial guess of λ_a is either 0 or the solution of the previous QPCC optimization when it is available (warm start).

E. Visualization

A by-product of the reduced formulation is that it allows one to visualize the problem when n_a is small (1,2 and partially for 3), even for complex contact configurations, with large n_c. This is helpful to better understand the properties of the problem, and it also has an educational value.

For a given I, the two last constraints of QP_{I}^r defines a polytope P_{I} (possibly empty or unbounded) in the space of λ_a. A face of this polytope corresponds to one line of these constraints holding as an equality, i.e. a change of complementarity choice. Since for every λ_a there is at least one I (and that when there are more than one, this corresponds precisely to changes of complementarity choices), the union of these polytopes cover the whole space of λ_a. Two polytopes P_{I_1} and P_{I_2} may share a part of their boundaries, when one goes from I_1 to I_2 by at most n_a changes (pivots).

Since for each λ_a there is a unique λ_c, we can express the cost function of the QPCC (7) as a function of λ_a only (for a given I, it is the cost function of QP_{I}). We denote as $c(\lambda_a)$ this function. It is defined by pieces, each piece support being a polytope P_{I}. With this representation, the original QPCC can be seen as an optimization problem with a piecewise-defined cost function with only the constraint on the actuation:

$$\begin{aligned} &\min_{\lambda_a}. \; c(\lambda_a) \\ &\text{s.t. } A\lambda_a \geq b \end{aligned} \tag{17}$$

When n_a is 1 or 2, it is possible to draw the graph of c. We give examples of such graphs for $n_a = 2$ on Fig. 2 and 3.

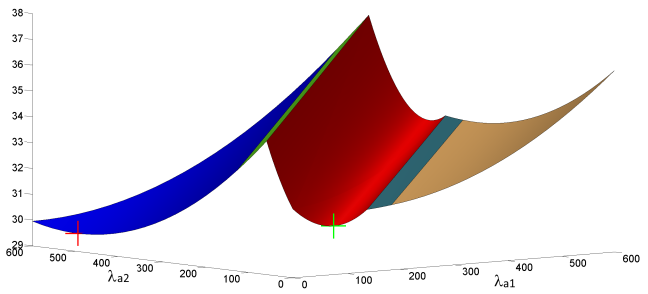

Fig. 2: Piecewise cost function for the problem corresponding to Fig. 4 (left), with $\lambda_a \in [0, 500]^2$. Each colored region correspond to a complementarity choice I and is supported by the corresponding polytope (polygon in the 2d case) P_{I}. There are two local minima corresponding to pulling one cable or the other. Pulling slightly the lower cable (up to $\lambda_{a1} \approx 75N$, local minimum with the green cross) lowers the beam but pulling more makes the end effector go up due to the lower contacts. Pulling on the upper cable makes use of the upper contacts to lower the end effector and reach the global minimum (red cross, $\lambda_{a2} \approx 495N$). One can see valleys along $\lambda_{a1} = \lambda_{a2}$. This is because the two cables are opposite, thus for a given λ_a, $\lambda_a + (\mu, \mu)^T$ ($\mu \in \mathbb{R}$) gives a similar (but not identical) end-effector displacement.

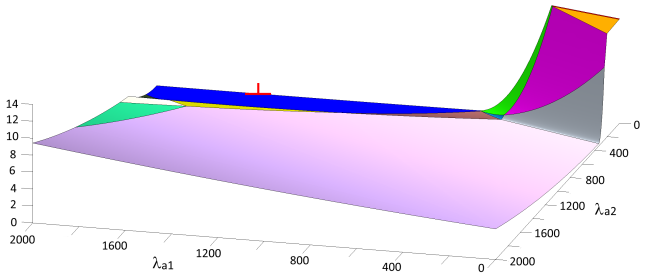

Fig. 3: An example for the same robot and different positions of obstacles, with $\lambda_a \in [0, 2000]^2$. Starting from $\lambda_a = (0, 0)$ (thin red zone in the upper right), our solver iterates across the regions in that order: red, orange, dark purple, green, cyan, brown, and finally blue, where the global minimum (red cross) is obtained. Note that the order to choose the pivot has an impact here: one could also have gone from the dark purple region to the gray one, and ended up in local minimum.

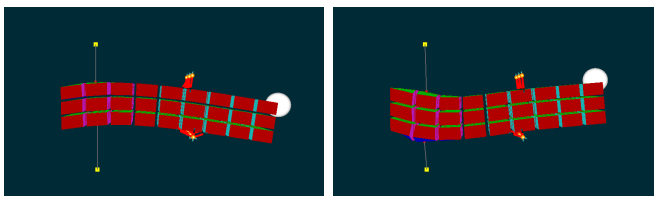

Fig. 4: Soft beam actuated with two cables. In this simulation we control the beam end-effector position. The orange spheres are fixed rigid obstacles. The white sphere is the desired position.

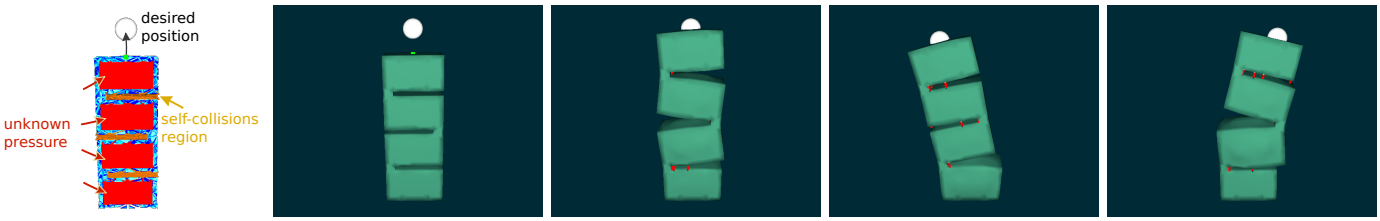

Fig. 5: Soft tower with four cavities actuated with pressure. The algorithm determines how to inflate each cavity to get the end-effector reach the desired position (white sphere). Self-collision points are represented by red lines.

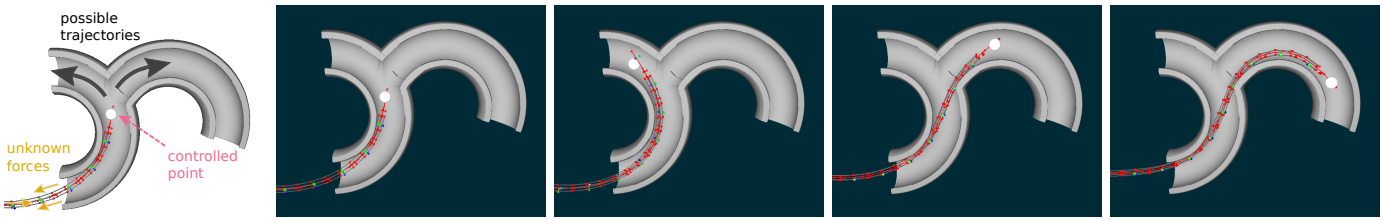

Fig. 6: Soft beam actuated with four cables. In this simulation we control the end-effector of a soft beam inserted in a pipe. The white sphere is the desired position.

IV. EXPERIMENTS AND RESULTS

A. Numerical examples

In this section we describe the results of our method on several numerical examples. We use qpOASES [19] to solve the QPs. In each case we control one point of the model and the target is interactively moved by a user or follows a predefined trajectory.

1) Tower: the first example is a simulation of a soft body with four cavities actuated with pressure (Fig. 5). Each inflatable section being separate from the others by a self-collision region. In this example, having the controlled point (top of the body) reaches its desired position entail the use of self-collisions. Note that, as the solver converges to a stationary point and not the global solution, the contacts have to be active ($\lambda_c > 0$) at the beginning of a QP resolution to be exploited by the actuation.

2) Rod: the second example is a simulation of a soft rod insertion into a pipe with a fork (Fig. 6). The rod has five actuators; four cables allow for orientation of the rod end-effector (up / down / left / right), and the fifth one moves the rod extremity along a fixed direction allowing its insertion into the pipe. Using our controller, we were able to optimize the five actuators so that the rod end-effector could be inserted in both branches. These results could be interesting, for instance, in interventional radiology, where therapeutic tools are inserted within the arteries through a catheter. We could imagine a simulation of the intervention, available in the operating room, and adapted to the patient's anatomy and physiology, to help the physician.

3) Beam: in Fig. 4 we show a simulation of a soft beam with one extremity fixed. The soft object is placed between two close obstacles that change the coupled direction between the actuators (two opposite cables) and the controlled point (tip of the beam). This simple example illustrates how the contacts are part of the optimization and can be used by the actuation to optimize the end-effector.

4) Two deformable bodies in contact: the last example is a simulation of two colliding soft bodies (Fig. 7), each body having some fixed part. The problem we want to solve in this inverse simulation is how to push the wall of a first deformable body, using an instrument, so that we can control a point of a second deformable body, thanks to transmission of forces allowed by contacts. In this example, we show that our method could have possible extension in medical applications (like medical robotics): Indeed, for some medical applications, it could be useful to know how much force/displacement to apply on a deformable wall (for instance with ultrasound probe when doing robotic assisted echography) in order to obtain a desired motion on an underlying deformable organ. Of course, dedicated study would be necessary to validate the use of the method in such an application. Here it just demonstrates the genericity of our algorithm and a potential larger use than soft robotics.

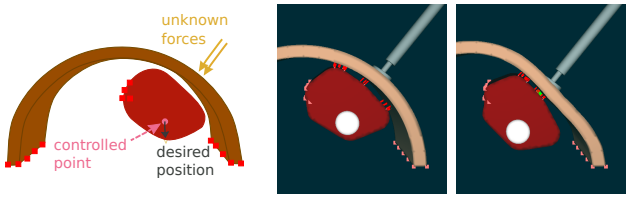

Fig. 7: Possible extension of the method: registration of colliding deformable bodies. The red squares are the fixed points. We control the position (white sphere) of a point of the red body by applying a force on the brown one.

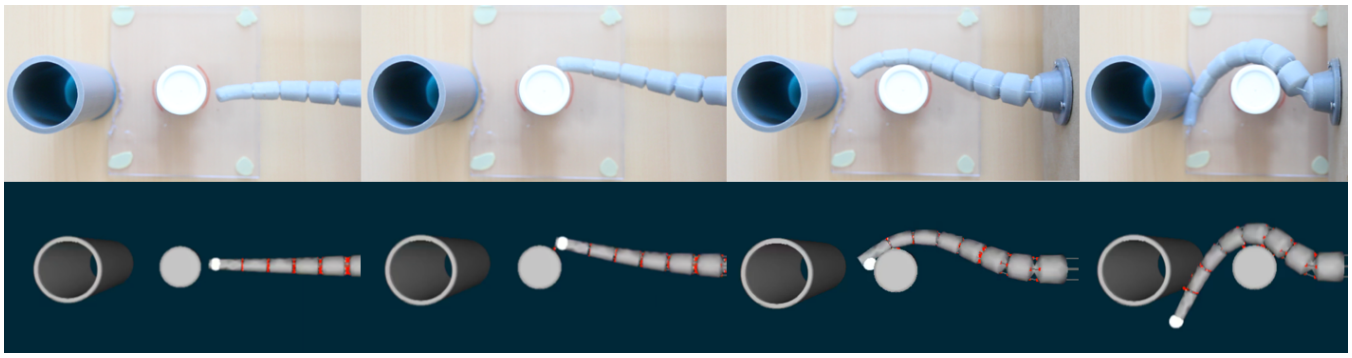

Fig. 8: Top: Real cable-driven soft robot, Bottom: Corresponding motion computed by the simulated inverse model. The input of the inverse model is the motion of the robot's tip.

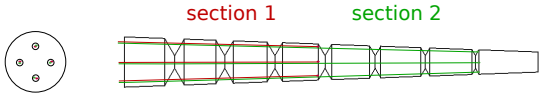

Fig. 9: Design of the soft trunk-like robot actuated with eight cables. A slice view (left) and a side view (right).

B. Real cable-driven soft robot

In this section we apply our method on a real cable-driven soft trunk-like robot made of silicone. In each experimental scenario we control the tip of the trunk and the target is interactively moved using a Gametrak device or follows a predefined trajectory.

The robot is actuated with eight cables. Four cables actuate a first section of the trunk while the other four go through the entire trunk (see Fig. 9), allowing it to perform a S-shape. In practice, we place flexible tubes inside the silicone that allows the cables to slip with low friction. To represent, in the simulation, the additional rigidity created by these tubes, we use a model of stiff springs in the direction of the tubes. The real trunk is attached to a platform moving along the robot direction, allowing a forward and backward displacement. This actuation is also modeled in the simulation. In this way, we were able to interactively drive the trunk end-effector between two cylinders using the optimization (see Fig. 8).

In Fig. 10, we show an example of the same robot but with a moving obstacle (a rigid cylinder). We used a Gametrak device to interactively update the position of the moving obstacle in the simulation. Using our algorithm we were able to find the new configurations of the robot actuation so that the end-effector keeps a fixed position. Having a real-time control of the robot motion could also be beneficial in scenarios where the robot actuation is changed, for example if a cable brakes. With feedback from the real robot we could interactively remove the broken cable from the simulation and get in real-time a new configuration for the actuators. In Fig. 11 we simulated this scenario, by interactively removing

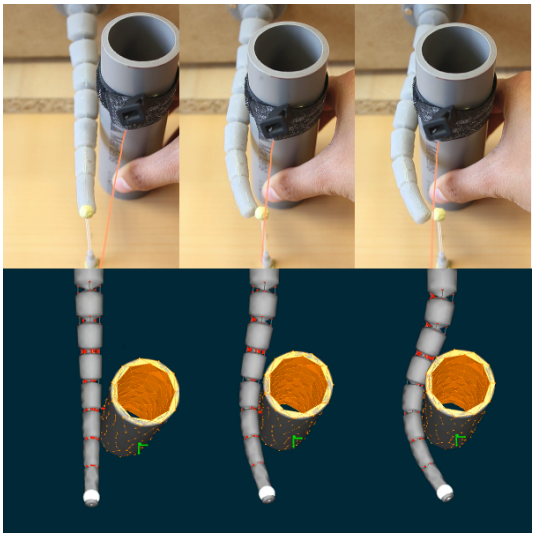

Fig. 10: Top: Real cable-driven soft robot, Bottom: Corresponding motion computed by the simulated inverse model. We interactively move an obstacle while the input of the inverse model is a fixed position of the robot's tip.

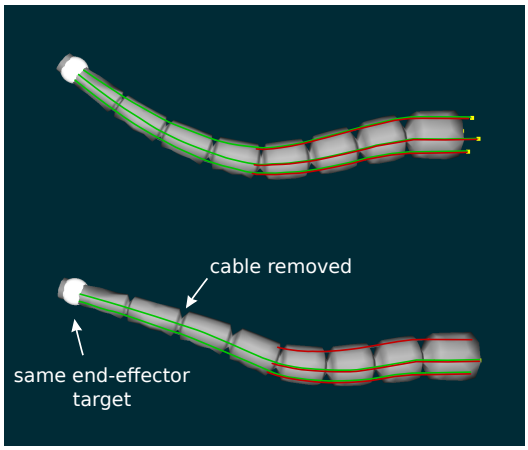

Fig. 11: Top: trunk-like robot inverse simulation. Bottom: same simulation, with same end-effector target, after interactively removed one cable. The algorithm finds a new configuration in real-time without losing the end-effector position.

one of the eight cables of the trunk-like robot (top one of the section 2, see Fig. 9). The algorithm was able to find a new configuration without losing the end-effector position.

C. Performance

In this section, we give the computation time[3] of each simulation shown in this paper. The two main computation steps of the simulation are the computation of the matrices W_{ij} $(i, j = e, a, c)$ and the resolution of the sequence of QP problems. In Table. 12, we show the average computation time for these two main steps. We can see that thanks to the projection of the model in the space of the actuation and contact variable, we are able to compute simulations at interactive rates (between 30Hz and 100Hz), and that the decomposition method we use to solve the contacts allows us to maintain these interactive rates. For more complex geometries, a large number of nodes may be required. In that case, the size of the FEM matrix will be large as well and the computation of $\mathbf{W}$ will take time. We use the work of [20] on asynchronous preconditioners to reduce this time and allow to compute in real-time, simulations of a robot with around 6000 DoFs.

Example	Model	Cont.	DoFs	W(ms)	QPs(ms)
Tower	FEM	21	1680	10.53	0.445
Rod	Beam	47	66	0.757	3.136
Beam	FEM	16	480	2.003	0.15
Two bodies	FEM	16	2313	10.458	0.123
Trunk	FEM	76	1665	23.288	5.529

Fig. 12: The different examples simulated, with their model used, the average number of contacts, the number of DoFs and the computation time in ms of the matrices W_{ij} construction and sequence of QPs resolution.

V. CONCLUSION AND FUTURE WORK

We proposed a generic method for interactive simulation and control of soft robots interacting with their environment. This method is based on a FEM simulation of the robot and QP with linear complementarity constraints. We were able to achieve fast inverse computation thanks to a reduced compliance matrix between controlled point, actuator and contact and dedicated optimization solver taking advantage of the problem properties. The method was applied to several numerical examples and on a real cable-driven soft robot.

This work focused on non frictional contact, but as a future work we will look forward on adding friction to the problem. We would also like to use this work for trajectory planning. Controlling the robot interactively can be interesting in applications such as assisting the medical intervention of catheter insertion. However, for other tasks like robot locomotion for instance, we may use the simulation as a tool for command planning. We are also looking forward to close the control loop with feedback information extracted from vision sensors. It would allow the robot to progress in a changing environment, and we could also use this feedback to correct errors introduced by the model.

[3]On a desktop computer with an i7 Intel processor 3.60GHz

ACKNOWLEDGMENT

The authors would like to thank Sanae Mahtal for her contribution on this project, and Mario Sanz Lopez for his precious help on building the trunk-like soft robot.

REFERENCES

[1] D. Trivedi, C. D. Rahn, W. M. Kier, and I. D. Walkerc, "Soft robotics: Biological inspiration, state of the art, and future research," *Applied Bionics and Biomechanics*, 2008.

[2] S. Kim, C. Laschi, and T. Barry, "Soft robotics: a bioinspired evolution in robotics," *Trends in biotechnology*, 2013.

[3] D. Rus and M. Tolley, "Design, fabrication and control of soft robots," *Nature*, 2015.

[4] R. Webster and B. Jones, "Design and kinematic modeling of constant curvature continuum robots: A review," *The International Journal of Robotics Research*, 2010.

[5] A. Marchese and D. Rus, "Design, kinematics, and control of a soft spatial fluidic elastomer manipulator," *The International Journal of Robotics Research*, 2016.

[6] C. Duriez, "Control of elastic soft robots based on real-time finite element method," *IEEE International Conference on Robotics and Automation*, 2013.

[7] F. Largilliere, V. Verona, E. Coevoet, M. Sanz-Lopez, J. Dequidt, and C. Duriez, "Real-time control of soft-robots using asynchronous finite element modeling," *IEEE International Conference on Robotics and Automation*, 2015.

[8] J. Kim and N. Pollard, "Direct control of simulated non-human characters," *IEEE Computer Graphics and Applicaions*, 2011.

[9] ——, "Fast simulation of skeleton-driven deformable body characters," *ACM Transaction on Graphics*, 2011.

[10] L. Liu, K. Yin, B. Wang, and B. Guo, "Simulation and control of skeleton-driven soft body characters," *ACM Transaction on Graphics (Proc. SIGGRAPH)*, 2013.

[11] J. Tan, G. Turk, and C. Liu, "Soft body locomotion," *ACM Transaction on Graphics (Proc. SIGGRAPH)*, 2012.

[12] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, and I. Peterlik, "Sofa: A multi-model framework for interactive physical simulation," *Studies in Mechanobiology Tissue Engineering and Biomaterials*, 2012.

[13] J. Hu, J. Mitchell, J. Pang, B. Yu, J. Hu, J. Mitchell, J. Pang, and B. Yu, "On linear programs with linear complementarity constraints," *Journal of Global Optimization*, 2012.

[14] L. Bai, J. Mitchell, and J. Pang, "On convex quadratic programs with linear complementarity constraints," *Computational Optimization and Applications*, 2013.

[15] L. Chen and D. Goldfarb, "An active set method for mathematical programs with linear complementarity constraints," *Computational Techniques and Applications*, 2007.

[16] G. Giallombardo and D. Ralph, "Multiplier convergence in trust-region methods with application to convergence of decomposition methods for mpecs," *Mathematical Programming*, 2008.

[17] G. Golub and C. Van Loan, *Matrix computations*, 3rd ed. John Hopkins University Press, 1996.

[18] K. G. Murty, "On the number of solutions of the complementarity problem and spanning properties of complementarity cones," *Linear Algebra and its Applications*, vol. 5.

[19] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, 2014.

[20] H. Courtecuisse, J. Allard, C. Duriez, and S. Cotin, "Asynchronous preconditioners for efficient solving of non-linear deformations," *Virtual Reality Interaction and Physical Simulation, Eurographics Association*, 2010.